Algorithms for Weighted Boolean Optimization

Vasco Manquinho[1], Joao Marques-Silva[2], Jordi Planes[3]

[1] IST/UTL - INESC-ID, vasco.manquinho@inesc-id.pt

[2] University College Dublin, jpms@ucd.ie

[3] Universitat de Lleida, jplanes@diei.udl.cat

March 6, 2009

Abstract

The Pseudo-Boolean Optimization (PBO) and Maximum Satisfiability (MaxSAT) problems are natural optimization extensions of Boolean Satisfiability (SAT). In the recent past, different algorithms have been proposed for PBO and for MaxSAT, despite the existence of straightforward mappings from PBO to MaxSAT and vice-versa. This papers proposes Weighted Boolean Optimization (WBO), a new unified framework that aggregates and extends PBO and MaxSAT. In addition, the paper proposes a new unsatisfiability-based algorithm for WBO, based on recent unsatisfiability-based algorithms for MaxSAT. Besides standard MaxSAT, the new algorithm can also be used to solve weighted MaxSAT and PBO, handling pseudo-Boolean constraints either natively or by translation to clausal form. Experimental results illustrate that unsatisfiability-based algorithms for MaxSAT can be orders of magnitude more efficient than existing dedicated algorithms. Finally, the paper illustrates how other algorithms for either PBO or MaxSAT can be extended to WBO.

1 Introduction

In the area of Boolean-based decision and optimization procedures, natural extensions of Boolean Satisfiability (SAT) include Maximum Satisfiability (MaxSAT) [10] and Pseudo-Boolean Optimization (PBO) [6]. Algorithms for MaxSAT and PBO have been the subject of significant improvements over the last few years. This in turn, motivated the use of both PBO and, more recently, of MaxSAT in a number of practical applications. Interestingly, albeit there are simple translations from any MaxSAT variant to PBO and vice-versa (by encoding to CNF) [1, 18], algorithms for MaxSAT and PBO have evolved separately, and often use fairly different algorithmic organizations. Nevertheless, there exists work that acknowledges this relationship and algorithms that can solve instances of MaxSAT and of PBO have already been proposed [1, 18].

Recent work has provided more alternatives for solving either MaxSAT or PBO, by using SAT solvers and the identification of unsatisfiable sub-formulas [16, 27]. However, the proposed algorithms were restricted to the plain and partial variants of MaxSAT and to a restricted form of Binate Covering for PBO. This paper extends this recent work in a number of directions. First, the paper proposes a simple algorithm for (Partial) Weighted MaxSAT, using unsatisfiable sub-formula identification. Second,

the paper generalizes MaxSAT and PBO by introducing Weighted Boolean Optimization (WBO), a new modeling framework for solving linear optimization problems over Boolean domains. Third, the paper shows how to extend the unsatisfiability-based algorithm for MaxSAT for solving WBO problems. Finally, the paper suggests how other algorithms can be used for solving WBO. Besides the proposed contributions, the paper also provides empirical evidence that unsatisfiability-based MaxSAT and WBO solvers can outperform state-of-the-art solvers on problem instances from practical problems.

The paper is organized as follows. Section 2 provides a brief overview of the topics addressed in the paper, namely MaxSAT, PBO, translations from MaxSAT to PBO and vice-versa, and unsatisfiability-based algorithms for MaxSAT. Section 3 details an algorithm for (Partial) Weighted MaxSAT based on unsatisfiable sub-formula identification. Next, Section 4 introduces Weighted Boolean Optimization (WBO), and shows how to extend the algorithm of Section 3 to WBO. Section 5 analyzes the experimental results, obtained on representative classes of problem instances. Section 6 overviews related work, and Section 7 concludes the paper.

2 Preliminaries

This section briefly introduces the Maximum Satisfiability (MaxSAT) problem and its variants, as well as the Pseudo-Boolean Optimization (PBO) problem. The main approaches used by state-of-the-art solvers are summarized. Moreover, translation procedures from MaxSAT to PBO and vice-versa are overviewed. Finally, unsatisfiability-based MaxSAT algorithms are surveyed, all of which the paper uses in later sections.

2.1 Maximum Satisfiability

Given a CNF formula φ, the Maximum Satisfiability (MaxSAT) problem can be defined as finding an assignment that maximizes the number of satisfied clauses (which implies that the assignment minimizes the number of unsatisfied clauses). Besides the classical MaxSAT problem, there are also three well-known variants of MaxSAT: weighted MaxSAT, partial MaxSAT and weighted partial MaxSAT. All these formulations have been used in a wide range of practical applications, namely scheduling, FPGA routing [34], design automation [31], among others.

A partial CNF formula is described as the conjunction of two CNF formulas φ_h and φ_s, where φ_h represents the *hard* clauses and φ_s represents the *soft* clauses. The *partial* MaxSAT problem consists in finding an assignment to the problem variables such that all hard clauses (φ_h) are satisfied, and the number of satisfied soft clauses (φ_s) is maximixed.

A weighted CNF formula is a set of weighted clauses. A weighted clause is a pair (ω, c), where ω is a classical clause and c is a natural number corresponding to the cost of unsatisfying ω. Given a weighted CNF formula, the *weighted* MaxSAT problem consists in finding an assignment to the problem variables such that the total weight of the satified clauses is maximized (which implies that the total weight of the unsatisfied clauses is minimized).

A weighted partial CNF formula is the conjunction of a weighted CNF formula (soft clauses) and a classical CNF formula (hard clauses). The *weighted partial* MaxSAT problem consists in finding an assignment to the variables such that all hard clauses are satisfied and the total weight of satisfied soft clauses is maximized. Observe that, for

both partial MaxSAT and weighted partial MaxSAT, hard clauses can also be represented as weighted clauses: one can consider that the weight is greater than the sum of the weights of the soft clauses.

Starting with the seminal work of Borchers and Furman [10], there has been an increasing interest in developing efficient MaxSAT solvers. Following such work, two branch and bound based solvers have been developed: (i) MaxSatz [20], the first solver to implement a unit propagation based lower bound and a failed literal based lower bound, both closely linked with a set of inference rules; (This solver has been extended into several solvers: IncMaxSatz [22], WMaxSatz [3], WMaxSatz_icss [13].) (ii) MiniMaxSAT [18], a solver created on top of MiniSAT with MaxSAT resolution [9] applied over an unsatisfiable sub-formula detected by the unit propagation based lower bound. A different approach has been the conversion of MaxSAT into a different formalism. The most notable works using this approach have been: Toolbar [19], a weighted CSP solver which converts MaxSAT instances into a weighted constraint network; SAT4J MAXSAT [7], a solver which iteratively converts a MaxSAT instance into a PBO instance; Clone [29] and sr(w) [30], solvers which convert a MaxSAT instance into a deterministic decomposable negation normal form (d-DNNF) instance; and MSUnCore [27], a solver which solves MaxSAT using the unsatisfiable cores detected by iteratively encoding the problem instance into SAT. In the Max-SAT Evaluations [4], this latter approach has been shown to be effective for industrial problems.

2.2 Pseudo-Boolean Optimization

The Pseudo-Boolean Optimization (PBO) problem is another extension of SAT where constraints can be any linear inequality with integer coefficients (also known as pseudo-Boolean constraints) defined over the set of problem variables. The objective in PBO is to find an assignment to problem variables such that all problem constraints are satisfied and the value of a linear objective function is optimized. Any pseudo-Boolean formulation can be easily translated into a normal form [6] such that all integer coefficients are non-negative.

$$\begin{array}{ll} \text{minimize} & \sum_{j \in N} c_j \cdot x_j \\ \text{subject to} & \sum_{j \in N} a_{ij} l_j \geq b_i, \\ & l_j \in \{x_j, \bar{x}_j\}, x_j \in \{0,1\}, a_{ij}, b_i, c_j \in \mathbf{N}_0^+ \end{array} \quad (1)$$

Almost all algorithms to solve PBO rely on the generalization of the most effective techniques already used in SAT solvers, namely Boolean Constraint Propagation, conflict-based learning and conflict-directed backtracking [24, 11]. Nevertheless, there are several approaches to solve PBO formulations. The most common using SAT solvers is to make a linear search on the value of the objective function. The idea is to generalize SAT algorithms to deal natively with pseudo-Boolean constraints [6] and whenever a solution for the problem constraints is found, a new constraint is added such that only solutions with a lower value for the objective function can be accepted. The algorithm finishes when the solver cannot improve on the last solution found, therefore proving its optimality.

Another common approach is branch and bound, where lower bounding procedures to estimate the value of the objective function are used. Several lower bounding procedures have been proposed, namely Maximum Independent Set of constraints [12], Linear Programming Relaxation [21, 23], among others [23]. There are also algorithms

that encode pseudo-Boolean constraints into propositional clauses [33, 5, 15] and solve the problem by subsequently using a SAT solver. This approach has been proved to be very effective for several problem sets, in particular when the clause encoding is not much larger than the original pseudo-Boolean formulation.

2.3 Translations between MaxSAT and PBO

Although MaxSAT and PBO are different formalisms, it is possible to encode any MaxSAT instance into a PBO instance and vice-versa [2, 1, 17]. This section focus solely on weighted partial MaxSAT, since the encodings of the other variants easily follow.

The encoding of hard clauses from weighted partial MaxSAT to PBO is straightforward, since propositional clauses are a particular case of pseudo-Boolean constraints. However, for each soft clause $\omega_i = (l_1 \vee l_2 \vee \ldots \vee l_k)$ with weight c_i, the encoding to PBO involves the use of an additional selection variable s_i, such that the corresponding constraint in PBO to ω_i would be $s_i + \sum_{j=1}^{k} l_j \geq 1$. This ensures that variable s_i is assigned to true whenever ω_i is not satisfied. The objective function of the corresponding PBO instance is to minimize the weighted sum of the selection variables. For each selection variable s_i in the objective function, its coefficient is the weight c_i of the corresponding soft clause ω_i.

Example. Consider the following weighted partial MaxSAT instance.

$$\begin{array}{ll} \varphi_h & = \{ \quad (x_1 \vee x_2 \vee \bar{x}_3), (\bar{x}_2 \vee x_3), (\bar{x}_1 \vee x_3) \} \\ \varphi_s & = \{ \quad (\bar{x}_3, 6), (x_1 \vee x_2, 3), (x_1 \vee x_3, 2) \} \end{array} \tag{2}$$

According to the described encoding, the corresponding PBO instance would be:

$$\begin{array}{rr} \text{minimize} & 6s_1 + 3s_2 + 2s_3 \\ \text{subject to} & x_1 + x_2 + \bar{x}_3 \geq 1 \\ & \bar{x}_2 + x_3 \geq 1 \\ & \bar{x}_1 + x_3 \geq 1 \\ & s_1 + \bar{x}_3 \geq 1 \\ & s_2 + x_1 + x_2 \geq 1 \\ & s_3 + x_1 + x_3 \geq 1 \end{array} \tag{3}$$

□

The encoding of PBO constraints into MaxSAT can be done using any of the proposed encodings from pseudo-Boolean constraints to clauses [33, 5, 15]. Hence, for each pseudo-Boolean constraint there will be a set of hard clauses encoding it in the respective MaxSAT instance. The number of clauses and additional variables, depends on the translation process used. The encoding is trivial when the original constraint in the PBO instance is already a clause.

The objective function of PBO instances can be encoded into MaxSAT with the use of weighted soft clauses. The idea is that for each variable x_j with coefficient c_j in the objective function, a corresponding soft clause $(\bar{x}_j)$ with weight c_j is added to the MaxSAT instance. Therefore, the solution of the MaxSAT formulation minimizes the weighted sum of problem variables, as required in the PBO instance.

Example. For illustration purposes, consider the following PBO instance:

$$\begin{array}{rr} \text{minimize} & 4x_1 + 2x_2 + x_3 \\ \text{subject to} & 2x_1 + 3x_2 + 5x_3 \geq 5 \\ & \bar{x}_1 + \bar{x}_2 \geq 1 \\ & x_1 + x_2 + x_3 \geq 2 \end{array} \tag{4}$$

Note that the first and third constraint must be encoded into CNF, but the second constraint is already a clause and so it can be represented directly as a hard clause. The corresponding MaxSAT instance would be:

$$\begin{array}{lll} \varphi_h & = \{ & \text{CNF}(2x_1 + 3x_2 + 5x_3 \geq 5), (\bar{x}_1 \vee \bar{x}_2), \text{CNF}(x_1 + x_2 + x_3 \geq 2)\} \\ \varphi_s & = \{ & (\bar{x}_1, 4), (\bar{x}_2, 2), (\bar{x}_3, 1)\} \end{array} \tag{5}$$

□

2.4 Unsatisfiability-Based MaxSAT

Recent work proposed the use of SAT solvers to solve (partial) MaxSAT, by iteratively identifying and relaxing unsatisfiable sub-formulas [16, 27, 26, 25]. In this paper we refer to these algorithms generically as MSU (Maximum Satisfiability with Unsatisfiability) algorithms.

The original algorithm of Fu&Malik (referred to as MSU1.0) iteratively identifies unsatisfiable sub-formulas. For each computed unsatisfiable sub-formula, all original (soft) clauses are relaxed with fresh relaxation variables. Moreover, a new Equals1 (or AtMost1) constraint relates the relaxation variables of each iteration, i.e. exactly 1 of these relaxation variables can be assigned value 1. The MSU1.0 algorithm can use more than one relaxation variables for each clause. In the original algorithm [16], a quadratic pairwise encoding of the Equals1 constraint was used. Finally, observe that the Equals1 constraint in line 13 of Algorithm 1 can be replaced by an AtMost1 constraint, without affecting the correctness of the algorithm.

More recently, several new MSU algorithms were proposed [26, 27]. The differences of the MSU algorithms include the number of cardinality constraints used, the encoding of cardinality constraints (of which the AtMost1 and Equals1 constraints are a special case), the number of relaxation variables considered for each clause, and how the MSU algorithm proceeds. Extensive experimentation (from [25] but also from the MaxSAT Evaluation [4]) suggests that an optimized variation of Fu&Malik's algorithm[25] is currently the best performing MSU algorithm.

3 Unsatisfiability-Based Weighted MaxSAT

This section describes extensions of MSU1.X, described in Algorithm 1, for solving (Partial) Weighted MaxSAT problems. One simple solution is to create c_j replicas of clause ω_j, where c_j is the weight of clause ω_j. The resulting extended CNF formula can then be solved by MSU1.X. The proof of Fu&Malik's paper would also apply in this case, and so correctness follows. The operation of this solution for (Partial) Weighted MaxSAT justifies a few observations. Consider an unsatisfiable sub-formula φ_C where the smallest weight is min_c. Each clause would be replaced by a number of replicas. Hence, this unsatisfiable sub-formula would be identified min_c times. Clearly, this solution is unlikely to scale for clauses with very large weights. Hence, a more effective solution is needed, which is detailed below.

An alternative solution is to split a clause only when the clause is included in an unsatisfiable sub-formula. The way the clause is split depends on its weight. An algorithm implementing this solution is shown in Algorithm 2. For each unsatisfiable sub-formula, the smallest weight min_c of the clauses in the sub-formula is computed.

Algorithm 1 The (Partial) MaxSAT algorithm of Fu&Malik [16]

```
MSU1(φ)
 1  φ_W ← φ                                          ▷ Working formula, initially set to φ
 2  while true
 3      do (st, φ_C) ← SAT(φ_W)
 4         ▷ φ_C is an unsatisfiable sub-formula if φ_W is unsat
 5         if st = UNSAT
 6           then V_R ← ∅
 7                for each ω ∈ φ_C
 8                    do if not hard(ω)
 9                         then r is a new relaxation variable
10                              ω_R ← ω ∪ {r}          ▷ ω_R is tagged non-auxiliary
11                              φ_W ← φ_W − {ω} ∪ {ω_R}
12                              V_R ← V_R ∪ {r}
13                φ_R ← CNF(∑_{r∈V_R} r = 1)          ▷ Equals1 constraint
14                Set all clauses in φ_R as hard clauses
15                φ_W ← φ_W ∪ φ_R                     ▷ Clauses in φ_R are declared hard
16           else ▷ Solution to MaxSAT problem
17                ν ← | blocking variables w/ value 1 |
18                return |φ| − ν
```

This smallest weight is then used to update a lower bound on minimum cost of unsatisfiable clauses. Clauses in the unsatisfiable sub-formula are relaxed. However, if the weight of a clause is larger than min_c, then the clause is split: a new relaxed clause with weight min_c is created, and the weight of the original clause is decreased by min_c.
Example. Consider the partial MaxSAT instance in (2). Assume that the unsatisfiable sub-formula detected in line 4 of Algorithm 2 is:

$$\varphi_C \quad = \quad \{\, (\bar{x}_2 \vee x_3), (\bar{x}_1 \vee x_3), \quad (\bar{x}_3, 6), (x_1 \vee x_2, 3)\,\}. \tag{6}$$

Then, the smallest weight min_c is 3, and the new formula becomes $\varphi_W = \varphi_h \cup \varphi_s$, where

$$\begin{array}{lll} \varphi_h & = \{ & (x_1 \vee x_2 \vee \bar{x}_3), (\bar{x}_2 \vee x_3), (\bar{x}_1 \vee x_3), \text{CNF}(s_1 + s_2 = 1)\,\} \\ \varphi_s & = \{ & (\bar{x}_3, 3), (x_1 \vee x_3, 2), (s_1 \vee \bar{x}_3, 3), (s_2 \vee x_1 \vee x_2, 3)\,\}. \end{array} \tag{7}$$

□

Observe that the new algorithm can be viewed as a direct optimization of the naive algorithm outlined earlier. The main difference is that each iteration of the algorithm collapses min_c iterations of the naive algorithm. For clauses with large weights the difference can be significant.
Theorem. [Correctness of WMSU1] The value returned by Algorithm 2 is minimum cost of non-satisfied clauses in φ. □

Proof. The previous discussion and the proof in [16]. □

4 Weighted Boolean Optimization

This section introduces Weighted Boolean Optimization (WBO), a new framework for modeling with hard and soft pseudo-Boolean constraints, that extends both MaxSAT

Algorithm 2 Unsatisfiability-based (Partial) Weighted MaxSAT algorithm

```
WMSU1(φ)
1  φ_W ← φ                                          ▷ Working formula, initially set to φ
2  cost_lb ← 0
3  while true
4      do (st, φ_C) ← SAT(φ_W)
5         ▷ φ_C is an unsatisfiable sub-formula if φ_W is unsat
6         if st = UNSAT
7           then min_c ← ∞
8                for each ω ∈ φ_C
9                    do if not hard(ω) and cost(ω) < min_c
10                        then min_c ← cost(ω)
11               cost_lb ← cost_lb + min_c
12               V_R ← ∅
13               for each ω ∈ φ_C
14                   do if not hard(ω)
15                       then r is a new relaxation variable
16                            V_R ← V_R ∪ {r}
17                            ω_R ← ω ∪ {r}            ▷ ω_R is tagged non-auxiliary
18                            cost(ω_R) ← min_c
19                            if cost(ω) > min_c
20                              then φ_W ← φ_W ∪ {ω_R}
21                                   cost(ω) ← cost(ω) − min_c
22                              else φ_W ← φ_W − {ω} ∪ {ω_R}
23               φ_R ← CNF(∑_{r∈V_R} r = 1)            ▷ Equals1 constraint
24               Set all clauses in φ_R as hard clauses
25               φ_W ← φ_W ∪ φ_R                      ▷ Clauses in φ_R are declared hard
26          else ▷ Solution to Weighted MaxSAT problem
27               return cost_lb
```

and its variants and PBO. Furthermore, a new algorithm based on identifying unsatisfiable sub-formulas is also proposed for solving WBO.

An Weighted Boolean Optimization (WBO) formula φ is composed of two sets of pseudo-Boolean constraints, φ_s and φ_h, where φ_s contains the soft constraints and φ_h contains the hard constraints. For each soft constraint $\omega_i \in \varphi_s$ there is an associated integer weight $c_i > 0$. The WBO problem consists in finding an assignment to the problem variables such that all hard constraints are satisfied and the total weight of the unsatisfied soft constraints is minimized (i.e. the total weight of satisfied soft constraints is maximized).

It should be noted that WBO represents a generalization of weighted partial MaxSAT by introducing the use of pseudo-Boolean constraints instead of just using propositional clauses. Hence, more compact formulations can be obtained with WBO than with MaxSAT. Moreover, PBO formulations can also be linearly encoded into WBO. Constraints in PBO can be directly encoded as hard constraints in WBO and the objective function can also be encoded as described in section 2.3. Therefore, WBO is a generalization of MaxSAT and its variants, as well as of PBO, allowing a unified modeling framework to integrate both of these Boolean optimization problems.

4.1 Unsatisfiability-Based WBO

This section describes how Algorithm 2 (introduced in Section 3) for weighted partial MaxSAT can be modified for solving WBO formulas. First of all, in a WBO formula, constraints are not restricted to be propositional clauses. Both soft and hard constraints can be pseudo-Boolean constraints. Hence, φ is a pseudo-Boolean formula, instead of a CNF formula. Moreover, the use of a SAT solver in line 6 is replaced with a pseudo-Boolean solver extended with the ability to generate an unsatisfiable sub-formula from the original pseudo-Boolean formula.

Next, if the formula is unsatisfiable, the weight associated with the unsatisfiable sub-formula is computed in the same way (lines 9-13) and the soft constraints in the core must also be relaxed using new relaxation variables (lines 15-24). Consider that $\omega = \sum a_j l_j \geq b$ denotes the pseudo-Boolean constraint to be relaxed using variable r. The resulting relaxed constraint in line 19 will be $\omega_R = b \cdot r + \sum a_j l_j \geq b$.

Finally, the constraint on the new relaxation variables in line 25 does not need to be encoded into CNF. The pseudo-Boolean constraint $\sum_{r \in V_R} r = 1$ can be directly added to φ_W, resulting in a more compact formulation, in particular if the number of soft constraints in the core is large.

In some cases, for an unsatisfiable sub-formula with k soft constraints, it is possible to use less than k additional variables. Consider the following soft constraints $\omega_1 = \sum_{l_j \in L_1} a_{1j} l_j \geq b_1$ and $\omega_2 = \sum_{l_j \in L_2} a_{2j} l_j \geq b_2$ in a given unsatisfiable sub-formula, where L_1 and L_2 denote respectively the set of literals in constraints ω_1 and ω_2. Additionally, let $x_k \in L_1$, $\bar{x}_k \in L_2$, $a_{1k} \geq b_1$ and $a_{2k} \geq b_2$, i.e. assigning x_k to true satisfies ω_1 and assigning x_k to false satisfies ω_2.[1] In this case, these constraints can share the same relaxing variable. This is due to the fact that it is impossible for both ω_1 and ω_2 to be unsatisfied by the same assignment, since either x_k satisfies ω_1 or $\bar{x}_k$ satisfies ω_2. Therefore, by using the same relaxing variable on both constraints, it is maintained the restriction that at most one soft constraint in the core can be relaxed.
Example. Suppose that the following set of soft constraints defines an unsatisfiable sub-formula in a WBO instance:

$$\begin{array}{rrl} \omega_1 = & 2x_1 + 3x_2 + 5x_3 & \geq 5 \\ \omega_2 = & \bar{x}_1 + \bar{x}_2 & \geq 1 \\ \omega_3 = & x_2 + \bar{x}_3 & \geq 1 \\ \omega_4 = & x_1 + \bar{x}_3 & \geq 1 \end{array} \tag{8}$$

In this case, constraints ω_1 and ω_3 can share the same relaxation variable, since the assignment of a value to x_3 implies that either ω_1 or ω_3 is satisfied. The same occurs with ω_2 and ω_4, given that the assignment to x_1 either satisfies ω_2 or ω_4. Therefore, after the relaxation, the resulting formula can include just two relaxation variables, instead of four. The resulting formula would be:

$$\begin{array}{rl} 5s_1 + 2x_1 + 3x_2 + 5x_3 & \geq 5 \\ s_2 + \bar{x}_1 + \bar{x}_2 & \geq 1 \\ s_1 + x_2 + \bar{x}_3 & \geq 1 \\ s_2 + x_1 + \bar{x}_3 & \geq 1 \\ s_1 + s_2 & \leq 1 \end{array} \tag{9}$$

□

[1] This is a generalization to pseudo-Boolean constraints. Note that if the WBO instance corresponds to a MaxSAT instance, this is very common to occur, since ω_1 and ω_2 are clauses.

Table 1: Classes of problem instances

Class	#I	MaxSAT Variant	Source
IND	110	Partial Weighted	To Appear in MaxSAT Evaluation 2009
FIR	59	Partial	Pseudo-Boolean Evaluation 2007
SYN	74	Partial	Pseudo-Boolean Evaluation 2005

The application of this reduction rule of relaxing variables raises the problem of finding the smallest number of relaxation variables to be used. This problem can be mapped into finding a matching of maximum cardinality in an undirected graph. In such a graph, there is a vertex for each constraint in the unsatisfiable sub-formula, while edges connect vertexes corresponding to constraints that can share a relaxation variable. The problem of finding a matching of maximum cardinality in an undirected graph can be solved in polynomial time [14]. Nevertheless, our prototype implementation of WBO solver uses a greedy algorithmic approach.

4.2 Other Algorithms for WBO

An alternative solution for solving WBO is to extend existing PBO algorithms. For example, soft pseudo-Boolean constraints can be represented in a PBO instance as relaxable constraints, and the overall cost function becomes the weighted sum of the relaxation variables of all soft pseudo-Boolean constraints of the original WBO formulation. This solution resembles the existing approach for solving MaxSAT with PBO [2, 1], and has the same potential drawbacks.

One additional alternative solution is to generalize branch and bound weighted partial MaxSAT solvers to deal with soft and hard pseudo-Boolean constraints. However, note that these approaches focus on a search process that uses successive refinements on the upper bound of the WBO solution, while the algorithm proposed in section 4.1 works by refining lower bounds on the optimum solution value.

5 Results

With the objective of evaluating the new (partial) weighted MaxSAT algorithm and the new WBO solver, a set of industrially-motivated problem instances was selected. The characteristics of the classes of instances considered are shown in Table 1. For each class of instances, the table provides the class name, the number of instances (#I), the type of MaxSAT variant, and the source for the class of instances.

Moreover, a wide range of MaxSAT and PBO solvers were considered, all among the best performing in either the MaxSAT or the Pseudo-Boolean evaluations. The weighted MaxSAT solvers considered were WMaxSatz [3], MiniMaxSat [18], IncW-MaxSatz [22], Clone [29], and SAT4J (MaxSAT) [7]. In addition, a new version of MSUnCore [26, 27, 25], integrating the weighted MaxSAT algorithm proposed in Section 3, was also evaluated. The PBO solvers considered were BSOLO [23], PBS [1], Pueblo [32], Minisat+[15], and SAT4J (PB) [7]. Finally, results for the new WBO solver, implementing the WBO organization described in Section 4 is also shown.

All experiments were run on a cluster of Linux AMD Opteron 2GHz servers with 1GB of RAM. The CPU time limit was set to 1800 seconds, and the RAM limit was set to 1 GB.

Table 2: Solved Instances for MaxSAT Solvers

Class	WMaxSatz	MiniMaxSat	IncWMaxSatz	Clone	SAT4J (MS)	MSUncore
IND	11	0	110	0	10	110
FIR	7	14	33	5	10	45
SYN	22	29	19	13	21	34
Total (Out of 243)	40	43	162	18	41	189

Table 3: Solved Instances for PBO & WBO Solvers

Class	BSOLO	PBS	Pueblo	Minisat+	SAT4J (PB)	WBO
IND	17	0	0	0	60	110
FIR	20	11	14	22	7	39
SYN	51	19	30	30	22	33
Total (Out of 243)	88	30	44	52	89	172

All algorithms were run on all problem instances considered. The original representations were used, in order to avoid introducing any bias towards any of the problem representations. Tables 2 and 3 summarize the number of instances aborted by each solver for each class of instances. As can be concluded, for practical problem instances, only a small number of MaxSAT solvers is effective. The results are somewhat different for the PBO solvers, where several can be competitive for different classes of instances. It should be noted that the IND benchmarks can be considered challenging for pseudo-Boolean solvers due to the large clause weights used.

For class IND and for the MaxSAT solvers, the results are somewhat surprising. Some of the solvers perform extremely well, whereas the others cannot solve most of the problem instances. IncWMaxSatz, MSUnCore and WBO are capable of solving *all* problem instances, but other MaxSAT solvers abort the vast majority of the problem instances. One additional observation is the very good performance of IncWMaxSatz when compared to WMaxSatz. This clearly indicates that the lower bound computation used in IncWMaxSatz can be very effective, even for industrial problem instances. For the PBO solvers, given the set of benchmark instances considered, SAT4J (PB) and BSOLO come out as the best performing. Clearly, this conclusion is based on the class of instances considered, which nevertheless derive from practical applications. Moreover, SAT4J (PB) performs significantly better than SAT4J (MaxSAT). This may be the result of a less effective encoding internally to SAT4J.

Motivated by the overall results, the best MaxSAT, PBO and the WBO solver were analyzed in more detail. Given the experimental results, IncWMaxSatz, MSUnCore, and WBO were selected. Figure 5 shows the results for the selected solvers by increasing run times.

As can be concluded, the plot confirms the trends in the tables of results. MSUnCore is the best performing, followed by WBO and IncWMaxSatz. For smaller run times (instances from class IND), IncWMaxSatz can be more efficient than WBO. Moreover, these results indicate that, for the classes of instances considered, encoding cardinality constraints into CNF (as done in MSUnCore) may be a better solution than natively handling cardinality and pseudo-Boolean constraints (as done in WBO). It

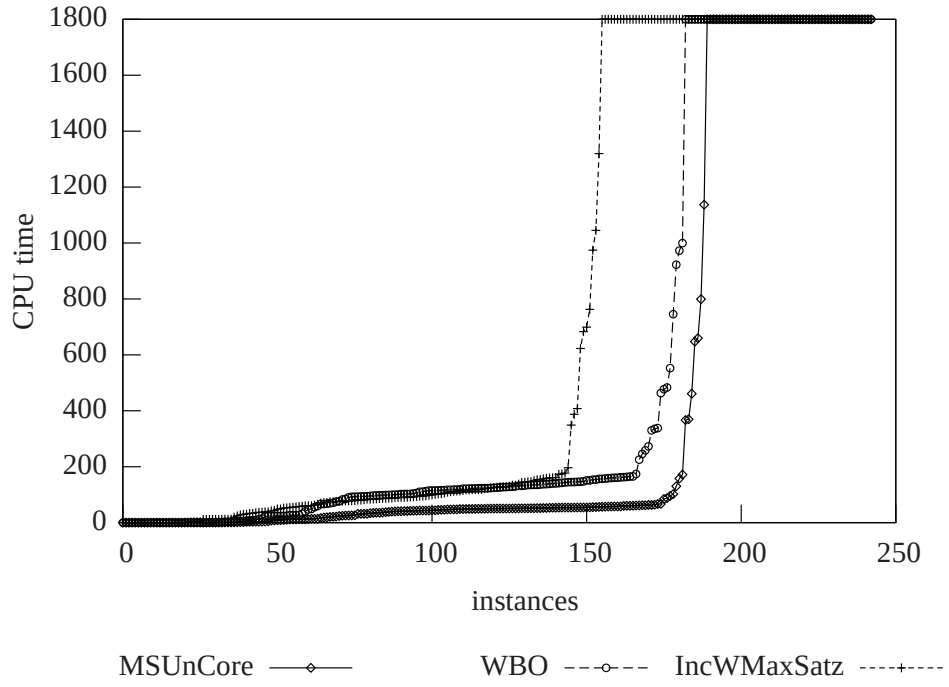

Figure 1: Run times for IncWMaxSatz, MSUnCore, and WBO for all instances

should be noted that all the instances considered can be encoded with cardinality constraints, for which existing polynomial encodings guarantee arc-consistency. This is not true for problem instances that use other pseudo-Boolean constraints, and for which encodings that ensure arc-consistency are exponential in the worst-case [15]. Finally, another source of difference in the experimental results is that whereas MSUnCore is built on top of PicoSAT [8], WBO is built on top of Minisat2. The different underlying SAT solvers may also contribute to explain some of the differences observed.

6 Related Work

A brief account of MaxSAT and PBO solvers is provided in Section 2. The use of unsatisfiability for solving MaxSAT was first proposed in 2006 [16]. This work was later extended [26, 27, 25], to accommodate several alternative algorithms and a number of optimizations to the first algorithm. To the best of our knowledge, MSUnCore is the first algorithm for solving (Partial) Weighted MaxSAT with unsatisfiable sub-formula identification. Also, to the best of our knowledge, WBO represents a new modeling framework, and the associate algorithm is new.

The use of optimization variants of decision procedures has also been proposed in the area of SMT [28], and a few SMT solvers now offer the ability for solving optimization problems. The approaches used for solving optimization problems in SMT are based on the use of relaxation variables, similarly to the PBO approach for solving MaxSAT [1].

7 Conclusions and Future Work

This paper proposes a new algorithm for (Partial) Weighted MaxSAT, based on unsatisfiable sub-formula identification. In addition, the paper introduces Weighted Boolean Optimization (WBO), that aggregates and generalizes PBO and MaxSAT. The paper then shows how unsatisfiability-based algorithms for (Partial) Weighted MaxSAT can be extended to WBO. Finally, the paper illustrates how to extend other algorithms for PBO and MaxSAT to solve WBO.

Experimental results, obtained on a representative set of benchmark instances shows that the new algorithm for weighted MaxSAT can outperform other existing algorithms by orders of magnitude. The experimental results also provide a preliminary (albeit possibly biased) study on the performance differences between handling pseudo-Boolean constraints natively and encoding to CNF. Finally, the paper shows that a general algorithm for WBO can be as efficient as other dedicated algorithms.

The integration of MaxSAT and PBO into a unique optimization extension of SAT increases the range of problems that can be solved. It also allows developing other general purpose algorithms, integrating the best techniques from both domains. Future research work will address adapting other algorithms for WBO. One concrete example is the use of PBO solvers. The other is extending the existing family of MSU algorithms for WBO.

Acknowledgement. This work is partially supported by EU grant ICT/217069 and FCT grant PTDC/EIA/76572/2006.

References

[1] F. Aloul, A. Ramani, I. Markov, and K. A. Sakallah. Generic ILP versus specialized 0-1 ILP: An update. In *International Conference on Computer-Aided Design*, pages 450–457, 2002.

[2] L. Amgoud, C. Cayrol, and D. L. Berre. Comparing arguments using preference ordering for argument-based reasoning. In *International Conference on Tools with Artificial Intelligence*, pages 400–403, 1996.

[3] J. Argelich, C. M. Li, and F. Manà. An improved exact solver for partial max-sat. In *Proceedings of the International Conference on Nonconvex Programming: Local and Global Approaches (NCP-2007)*, pages 230–231, 2007.

[4] J. Argelich, C. M. Li, F. Manyà, and J. Planes. Third Max-SAT evaluation. www.maxsat.udl.cat/08/, 2008.

[5] O. Bailleux, Y. Boufkhad, and O. Roussel. A translation of pseudo Boolean constraints to SAT. *Journal on Satisfiability, Boolean Modeling and Computation*, 2:191–200, 2006.

[6] P. Barth. A Davis-Putnam Enumeration Algorithm for Linear Pseudo-Boolean Optimization. Technical Report MPI-I-95-2-003, Max Plank Institute for Computer Science, 1995.

[7] D. L. Berre. SAT4J library. www.sat4j.org.

[8] A. Biere. PicoSAT essentials. *Journal on Satisfiability, Boolean Modeling and Computation*, 2:75–97, 2008.

[9] M. L. Bonet, J. Levy, and F. Manyà. Resolution for Max-SAT. *Artificial Intelligence*, 171(8–9):606–618, 2007.

[10] B. Borchers and J. Furman. A two-phase exact algorithm for MAX-SAT and weighted MAX-S AT problems. *Journal of Combinatorial Optimization*, 2:299–306, 1999.

[11] D. Chai and A. Kuehlmann. A fast pseudo-Boolean constraint solver. In *Design Automation Conference*, pages 830–835, 2003.

[12] O. Coudert. On Solving Covering Problems. In *Proceedings of the Design Automation Conference*, pages 197–202, 1996.

[13] S. Darras, G. Dequen, L. Devendevill, and C. M. Li. On inconsistent clause-subsets for Max-SAT solving. In *CP-07*, volume 4741 of *LNCS*, pages 225–240. Springer, 2007.

[14] J. Edmonds. Paths, trees and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[15] N. Een and N. Sörensson. Translating pseudo-Boolean constraints into SAT. *Journal on Satisfiability, Boolean Modeling and Computation*, 2, 2006.

[16] Z. Fu and S. Malik. On solving the partial MAX-SAT problem. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 252–265, August 2006.

[17] F. Heras, J. Larrosa, and A. Oliveras. MiniMaxSat: a new weighted Max-SAT solver. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 41–55, May 2007.

[18] F. Heras, J. Larrosa, and A. Oliveras. MiniMaxSAT: An efficient weighted Max-SAT solver. *Journal of Artificial Intelligence Research*, 31:1–32, 2008.

[19] J. Larrosa, F. Heras, and S. de Givry. A logical approach to efficient Max-SAT solving. *Artificial Intelligence*, 172(2-3):204–233, 2008.

[20] C. M. Li, F. Manyà, and J. Planes. New inference rules for Max-SAT. *Journal of Artificial Intelligence Research*, 30:321–359, 2007.

[21] S. Liao and S. Devadas. Solving Covering Problems Using LPR-Based Lower Bounds. In *Proceedings of the Design Automation Conference*, pages 117–120, 1997.

[22] H. Lin and K. Su. Exploiting inference rules to compute lower bounds for MAX-SAT solving. In *IJCAI-07*, pages 2334–2339, 2007.

[23] V. Manquinho and J. Marques-Silva. Effective lower bounding techniques for pseudo-boolean optimization. In *Design, Automation and Test in Europe Conference*, pages 660–665. ACM Press, 2005.

[24] V. M. Manquinho and J. Marques-Silva. Search pruning techniques in SAT-based branch-and-bound algorithms for the binate covering problem. *IEEE Transactions on Computer-Aided Design*, 21(5):505–516, 2002.

[25] J. Marques-Silva and V. Manquinho. Towards more effective unsatisfiability-based maximum satisfiability algorithms. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 225–230, March 2008.

[26] J. Marques-Silva and J. Planes. On using unsatisfiability for solving maximum satisfiability. *Computing Research Repository*, abs/0712.0097, December 2007.

[27] J. Marques-Silva and J. Planes. Algorithms for maximum satisfiability using unsatisfiable cores. In *Design, Automation and Testing in Europe Conference*, pages 408–413, March 2008.

[28] R. Nieuwenhuis and A. Oliveras. On SAT modulo theories and optimization problems. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 156–169, 2006.

[29] K. Pipatsrisawat, A. Palyan, M. Chavira, A. Choi, and A. Darwiche. Solving weighted Max-SAT problems in a reduced search space: A performance analysis. *Journal on Satisfiability Boolean Modeling and Computation (JSAT)*, 4:191–217, 2008.

[30] M. Ramírez and H. Geffner. Structural relaxations by variable renaming and their compilation for solving MinCostSAT. In C. Bessiere, editor, *The 13th International Conference on Principles and Practice of Constraint Programming*, volume 4741 of *LNCS*, pages 605–619. Springer, 2007.

[31] S. Safarpour, H. Mangassarian, A. Veneris, M. H. Liffiton, and K. A. Sakallah. Improved design debugging using maximum satisfiability. In *Formal Methods in Computer-Aided Design*, 2007.

[32] H. Sheini and K. A. Sakallah. Pueblo: A hybrid pseudo-Boolean SAT solver. *Journal on Satisfiability, Boolean Modeling and Computation*, 2:165–189, 2006.

[33] J. P. Warners. A linear-time transformation of linear inequalities into conjunctive normal form. *Information Processing Letters*, 68(2):63–69, 1998.

[34] H. Xu, R. A. Rutenbar, and K. A. Sakallah. sub-SAT: a formulation for relaxed boolean satisfiability with applications in routing. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 22(6):814–820, 2003.